```php
<?php

<?php

/**
* Функции вывода HTML
*
* @license     GPL 2 ( http://www.gnu.org/licenses/gpl.html)
* @автор   Андреас Гор <andi@splitbrain.org>
*/

use dokuwiki\Ui\MediaRevisions;
use dokuwiki\Form\Form;
use dokuwiki\Action\Denied;
use dokuwiki\Action\Locked;
use dokuwiki\ChangeLog\PageChangeLog;
use dokuwiki\Extension\AuthPlugin;
use dokuwiki\Extension\Event;
use dokuwiki\Ui\Backlinks;
use dokuwiki\Ui\Editor;
use dokuwiki\Ui\Index;
use dokuwiki\Ui\Login;
use dokuwiki\Ui\PageConflict;
use dokuwiki\Ui\PageDiff;
use dokuwiki\Ui\PageDraft;
use dokuwiki\Ui\PageRevisions;
use dokuwiki\Ui\PageView;
use dokuwiki\Ui\Recent;
use dokuwiki\Ui\UserProfile;
use dokuwiki\Ui\UserRegister;
use dokuwiki\Ui\UserResendPwd;
use dokuwiki\Utf8\Clean;

if (!defined('SEC_EDIT_PATTERN')) {
    define('SEC_EDIT_PATTERN', '#<!-- EDIT({.*?}) -->#');
}


/**
* Удобная функция для быстрого создания вики-ссылки
*
* @автор Андреас Гор <andi@splitbrain.org>
* @param  string   $ id       идентификатор целевой страницы
* @param  string   $ name     имя ссылки, т.е. текст, который отображается
* @param  string | array   $ search    строка(ы) поиска, которая должна быть
выделена на целевой странице
* @return  string HTML-код ссылки
*/
function html_wikilink($id, $name = null, $search = '')
{
    /** @var Doku_Renderer_xhtml $xhtml_renderer */
```

```
49      static $xhtml_renderer = null;
50      if (is_null($xhtml_renderer)) {
51          $xhtml_renderer = p_get_renderer('xhtml');
52      }
53
54      return $xhtml_renderer->internallink($id, $name, $search, true,
'navigation');
55  }
56
57  /**
58   * Форма входа
59   *
60   * @автор  Андреас Гор <andi@splitbrain.org>
61   *
62   * @param  bool $ svg Показывать ли иконки svg в ссылках register и resendpwd
или нет
63   * @устаревший 2020-07-18
64   */
65  function html_login($svg = false)
66  {
67      dbg_deprecated(Login::class . '::show()');
68      (new Login($svg))->show();
69  }
70
71
72  /**
73   * Отклоненное содержимое страницы
74   *
75   * @deprecated 2020-07-18 больше не вызывается, см. inc / Действие / Отклонено
::tplContent()
76   */
77  function html_denied()
78  {
79      dbg_deprecated(Denied::class . '::showBanner()');
80      (new Denied())->showBanner();
81  }
82
83  /**
84   * вставляет кнопки редактирования раздела, если необходимо, или удаляет маркеры
85   *
86   * @автор Андреас Гор <andi@splitbrain.org>
87   *
88   * @param  строка $ текст
89   * @param  bool    $ show показать кнопки редактирования раздела?
90   * @возвращаемая строка
91   */
92  function html_secedit($text, $show = true)
93  {
94      global $INFO;
95
96      if ((isset($INFO) && !$INFO['writable']) || !$show || (isset($INFO)
```

```
&& $INFO['rev'])) {
97          return preg_replace(SEC_EDIT_PATTERN, '', $text);
98      }
99
100     return preg_replace_callback(
101         SEC_EDIT_PATTERN,
102         'html_secedit_button',
103         $text
104     );
105 }
106
107/**
108* подготавливает данные кнопки редактирования раздела для запуска события
109* используется как обратный вызов в  html_secedit
110*
111* @автор Андреас Гор <andi@splitbrain.org>
112*
113* @param  array $ соответствует совпадениям с регулярным выражением
114* @возвращаемая строка
115* @triggers HTML_SECEDIT_BUTTON
116*/
117 function html_secedit_button($matches)
118 {
119     $json = htmlspecialchars_decode($matches[1], ENT_QUOTES);
120
121     try {
122         $data = json_decode($json, true, 512, JSON_THROW_ON_ERROR);
123     } catch (JsonException $e) {
124         return '';
125     }
126     $data['target'] = strtolower($data['target']);
127     $data['hid'] = strtolower($data['hid'] ?? '');
128
129     return Event::createAndTrigger(
130         'HTML_SECEDIT_BUTTON',
131         $data,
132         'html_secedit_get_button'
133     );
134 }
135
136/**
137* печатает кнопку редактирования раздела
138* используется как действие по умолчанию формы  HTML_SECEDIT_BUTTON
139*
140* @автор Адриан Лэнг <lang@cosmocode.de>
141*
142* @param  array $ имя данных , идентификатор раздела и цель
143* @return  string html
144*/
145 function html_secedit_get_button($data)
146 {
```

```
147     global $ID;
148     global $INFO;
149 
150     if (!isset($data['name']) || $data['name'] === '') return '';
151 
152     $name = $data['name'];
153     unset($data['name']);
154 
155     $secid = $data['secid'];
156     unset($data['secid']);
157 
158     $params = array_merge(
159         ['do'  => 'edit', 'rev' => $INFO['lastmod'], 'summary' => '[' .
$name . '] '],
160         $data
161     );
162 
163     $html = '<div class="secedit editbutton_' . $data['target'] . '
editbutton_' . $secid . '">';
164     $html .= html_btn('secedit', $ID, '', $params, 'post', $name);
165     $html .= '</div>';
166     return $html;
167 }
168 
169/**
170* Только кнопка «Наверх» (в своей собственной форме)
171*
172* @автор Андреас Гор <andi@splitbrain.org>
173*
174* @return  string html
175*/
176 function html_topbtn()
177 {
178     global $lang;
179 
180     return '<a class="nolink" href="#dokuwiki__top">'
181         . '<button class="button" onclick="window.scrollTo(0, 0)"
title="' . $lang['btn_top'] . '">'
182         . $lang['btn_top']
183         . '</button></a>';
184 }
185 
186/**
187* Отображает кнопку (используя собственную форму)
188* Если подсказка существует, подсказка клавиши доступа заменяется.
189*
190* @автор Андреас Гор <andi@splitbrain.org>
191*
192  * @param string         $name
193  * @param string         $id
194  * @param string         $akey   ключ доступа
```

```
195  * @param string[]       $params key-value параметр "ключ-значение" добавлены как скрытые входные данные
196  * @param string         $method  метод
197  * @param string         $tooltip  подсказка
198  * @param bool|string    $label  текст метки, false: поиск btn_$name в локализации
199  * @param string         $svg (необязательно) код svg, вставленный в кнопку
200  * @return string          возвращаемая строка
201  */
202 function html_btn($name, $id, $akey, $params, $method = 'get', $tooltip = '', $label = false, $svg = null)
203 {
204     global $conf;
205     global $lang;
206
207     if (!$label)
208         $label = $lang['btn_' . $name];
209
210     //идентификатор фильтра (без URL-кодирования)
211     $id = idfilter($id, false);
212
213     //создайте красивые URL-адреса даже для кнопок
214     if ($conf['userewrite'] == 2) {
215         $script = DOKU_BASE . DOKU_SCRIPT . '/' . $id;
216     } elseif ($conf['userewrite']) {
217         $script = DOKU_BASE . $id;
218     } else {
219         $script = DOKU_BASE . DOKU_SCRIPT;
220         $params['id'] = $id;
221     }
222
223     $html = '<form class="button btn_' . $name . '" method="' . $method . '" action="' . $script . '"><div class="no">';
224
225     if (is_array($params)) {
226         foreach ($params as $key => $val) {
227             $html .= '<input type="hidden" name="' . $key . '" value="' . hsc($val) . '" />';
228         }
229     }
230
231     $tip = empty($tooltip) ? hsc($label) : hsc($tooltip);
232
233     $html .= '<button type="submit" ';
234     if ($akey) {
235         $tip  .= ' [' . strtoupper($akey) . ']';
236         $html .= 'accesskey="' . $akey . '" ';
237     }
238     $html .= 'title="' . $tip . '">';
239     if ($svg) {
240         $html .= '<span>' . hsc($label) . '</span>' . inlineSVG($svg);
```

```
241     } else {
242         $html .= hsc($label);
243     }
244     $html .= '</button>';
245     $html .= '</div></form>';
246 
247     return $html;
248 }
249/**
250* показать предупреждение о пересмотре
251*
252* @author Шимон Олевничак <dokuwiki@imz.re>
253* @устаревший 2020-07-18
254*/
255 function html_showrev()
256 {
257     dbg_deprecated(PageView::class . '::showrev()');
258 }
259 
260/**
261* Показать вики-страницу
262*
263* @автор Андреас Гор <andi@splitbrain.org>
264*
265* @param  null | string $ txt wiki-текст или null для отображения $ID
266* @устаревший 2020-07-18
267*/
268 function html_show($txt = null)
269 {
270     dbg_deprecated(PageView::class . '::show()');
271     (new PageView($txt))->show();
272 }
273 
274/**
275* спросить пользователя о том, как обращаться с существующим черновиком
276*
277* @автор Андреас Гор <andi@splitbrain.org>
278* @устаревший 2020-07-18
279*/
280 function html_draft()
281 {
282     dbg_deprecated(PageDraft::class . '::show()');
283     (new PageDraft())->show();
284 }
285 
286/**
287* Выделяет поисковые запросы в HTML-коде
288*
289* @автор Андреас Гор <andi@splitbrain.org>
290* @автор Гарри Фьюкс <hfuecks@gmail.com>
291*
```

```
292* @param  string $ html
293* @param  массив | строка $ фразы
294* @return  string html
295*/
296 function html_hilight($html, $phrases)
297 {
298     $phrases = (array) $phrases;
299     $phrases = array_map('preg_quote_cb', $phrases);
300     $phrases = array_map('ft_snippet_re_preprocess', $phrases);
301     $phrases = array_filter($phrases);
302
303     $regex = implode('|', $phrases);
304
305     if ($regex === '') return $html;
306     if (!Clean::isUtf8($regex)) return $html;
307
308     return @preg_replace_callback("/((<[^>]*)|$regex)/ui", function
($match) {
309         $hlight = unslash($match[0]);
310         if (!isset($match[2])) {
311             $hlight = '<span class="search_hit">' . $hlight . '</span>';
312         }
313         return $hlight;
314     }, $html);
315 }
316
317/**
318* Отображение ошибки на заблокированных страницах
319*
320* @автор Андреас Гор <andi@splitbrain.org>
321* @deprecated 2020-07-18 больше не вызывается, см. inc / Действие /
Заблокировано ::tplContent()
322*/
323 function html_locked()
324 {
325     dbg_deprecated(Locked::class . '::showBanner()');
326     (new Locked())->showBanner();
327 }
328
329/**
330* список старых ревизий
331*
332* @автор Андреас Гор <andi@splitbrain.org>
333* @автор Бен Кобурн <btcoburn@silicodon.net>
334* @author Катя Арзамасцева <pshns@ukr.net>
335*
336* @param  int $ сначала пропустить первые n строк журнала изменений
337* @param  string $ media_id идентификатор носителя или пусто для текущей
страницы
338* @устаревший 2020-07-18
339*/
```

```php
function html_revisions($first = -1, $media_id = '')
{
    dbg_deprecated(PageRevisions::class . '::show()');
    if ($media_id) {
        (new MediaRevisions($media_id))->show($first);
    } else {
        global $INFO;
        (new PageRevisions($INFO['id']))->show($first);
    }
}

/**
* показать последние изменения
*
* @автор Андреас Гор <andi@splitbrain.org>
* @автор Маттиас Гримм <matthiasgrimm@users.sourceforge.net>
* @автор Бен Кобурн <btcoburn@silicodon.net>
* @author Катя Арзамасцева <pshns@ukr.net>
*
* @param  int $ первый
* @параметр строка $ show_changes
* @устаревший 2020-07-18
*/
function html_recent($first = 0, $show_changes = 'both')
{
    dbg_deprecated(Recent::class . '::show()');
    (new Recent($first, $show_changes))->show();
}

/**
* Показать индекс страницы
*
* @автор Андреас Гор <andi@splitbrain.org>
*
* @param  string $ ns
* @устаревший 2020-07-18
*/
function html_index($ns)
{
    dbg_deprecated(Index::class . '::show()');
    (new Index($ns))->show();
}

/**
* Форматировщик элементов дерева индекса для html_buildlist()
*
* Пользовательская функция для html_buildlist()
*
* @автор Андреас Гор <andi@splitbrain.org>
*
* @param  массив $ элемент
```

```
391* @возвращаемая строка
392* @устаревший 2020-07-18
393*/
394 function html_list_index($item)
395 {
396     dbg_deprecated(Index::class . '::formatListItem()');
397     return (new Index())->formatListItem($item);
398 }
399
400/**
401* Форматировщик элементов списка индексов для html_buildlist()
402*
403* Эта пользовательская функция используется в html_buildlist для построения
404* Теги <li> для пространств имен при отображении индекса страницы
405* присваивает различные классы открытым и закрытым «папкам»
406*
407* @автор Андреас Гор <andi@splitbrain.org>
408*
409* @param  массив $ элемент
410* @return  string html
411* @устаревший 2020-07-18
412*/
413 function html_li_index($item)
414 {
415     dbg_deprecated(Index::class . '::tagListItem()');
416     return (new Index())->tagListItem($item);
417 }
418
419/**
420* Форматировщик элементов списка по умолчанию для html_buildlist()
421*
422* @автор Андреас Гор <andi@splitbrain.org>
423*
424* @param  массив $ элемент
425* @return  string html
426* @устаревший 2020-07-18
427*/
428 function html_li_default($item)
429 {
430     return '<li class="level' . $item['level'] . '">';
431 }
432
433/**
434* Создайте неупорядоченный список
435*
436* Создать неупорядоченный список из заданного массива $data
437* Каждый элемент массива должен иметь свойство «уровень»
438* сам элемент печатается указанным пользователем $func
439* функция. Вторая и необязательная функция используется для
440* распечатать тег <li>. Обе пользовательские функции должны принять
441* один предмет.
```

```
442*
443* Обе пользовательские функции могут быть заданы как массив для указания
444* член объекта.
445*
446* @автор Андреас Гор <andi@splitbrain.org>
447*
448* @param  array     $ массив данных  с массивами элементов
449* @param  string    $ class класс оболочки ul
450* @param  callable $ func   callback для печати элемента списка
451* @param  callable $ lifunc (необязательно) обратный вызов открывающего тега
li
452* @param  bool      $ forcewrapper (необязательно) Инициирует создание
обертки ul, если первый уровень равен
453* 0 (у нас есть корневой объект) или 1 (только корневое содержимое)
454* @return  string html неупорядоченного списка
455*/
456 function html_buildlist($data, $class, $func, $lifunc = null,
$forcewrapper = false)
457 {
458     if ($data === []) {
459         return '';
460     }
461
462     $firstElement = reset($data);
463     $start_level = $firstElement['level'];
464     $level = $start_level;
465     $html  = '';
466     $open  = 0;
467
468     // set callback function to build the <li> tag, formerly defined as
html_li_default()
469     if (!is_callable($lifunc)) {
470         $lifunc = static fn($item) => '<li class="level' .
$item['level'] . '">';
471     }
472
473     foreach ($data as $item) {
474         if ($item['level'] > $level) {
475             //open new list
476             for ($i = 0; $i < ($item['level'] - $level); $i++) {
477                 if ($i) $html .= '<li class="clear">';
478                 $html .= "\n" . '<ul class="' . $class . '">' . "\n";
479                 $open++;
480             }
481             $level = $item['level'];
482         } elseif ($item['level'] < $level) {
483             //close last item
484             $html .= '</li>' . "\n";
485             while ($level > $item['level'] && $open > 0) {
486                 //close higher lists
487                 $html .= '</ul>' . "\n" . '</li>' . "\n";
```

```
488                 $level--;
489                 $open--;
490             }
491         } elseif ($html !== '') {
492             //close previous item
493             $html .= '</li>' . "\n";
494         }
495 
496         //print item
497         $html .= call_user_func($lifunc, $item);
498         $html .= '<div class="li">';
499 
500         $html .= call_user_func($func, $item);
501         $html .= '</div>';
502     }
503 
504     //close remaining items and lists
505     $html .= '</li>' . "\n";
506     while ($open-- > 0) {
507         $html .= '</ul></li>' . "\n";
508     }
509 
510     if ($forcewrapper || $start_level < 2) {
511         // Trigger building a wrapper ul if the first level is
512         // 0 (we have a root object) or 1 (just the root content)
513         $html = "\n" . '<ul class="' . $class . '">' . "\n" . $html .
'</ul>' . "\n";
514     }
515 
516     return $html;
517 }
518 
519/**
520* отображать обратные ссылки
521*
522* @автор Андреас Гор <andi@splitbrain.org>
523* @author Майкл Клиер <chi@chimeric.de>
524* @устаревший 2020-07-18
525*/
526 function html_backlinks()
527 {
528     dbg_deprecated(Backlinks::class . '::show()');
529     (new Backlinks())->show();
530 }
531 
532/**
533* Получить заголовок diff HTML
534*
535* @param  string $ l_rev    Оставшиеся ревизии
536* @param  string $ r_rev    Правая ревизия
537* @param  string $ id       Идентификатор страницы, если используется null
```

```
$ID
538* @param  bool    $ media    Если это для медиа-файлов
539* @param  bool    $ inline   Возвращает заголовок в одной строке
540* @возврат строки []HTML-фрагменты для заголовка diff
541* @устаревший 2020-07-18
542*/
543 function html_diff_head($l_rev, $r_rev, $id = null, $media = false,
$inline = false)
544 {
545     dbg_deprecated('see ' . PageDiff::class . '::buildDiffHead()');
546     return ['', '', '', ''];
547 }
548
549/**
550* Показать разницу
551* между текущей версией страницы и предоставленным $text
552* или между ревизиями, предоставленными через GET или POST
553*
554* @автор Андреас Гор <andi@splitbrain.org>
555* @param   string $ text   если непустой: сравнить этот текст с самой последней
версией
556* @param   bool    $ intro отобразить вступительный текст
557* @param   string $ type   тип diff (встроенный или параллельный)
558* @устаревший 2020-07-18
559*/
560 function html_diff($text = '', $intro = true, $type = null)
561 {
562     dbg_deprecated(PageDiff::class . '::show()');
563     global $INFO;
564     (new PageDiff($INFO['id']))->compareWith($text)->preference([
565         'showIntro' => $intro,
566         'difftype'  => $type,
567     ])->show();
568 }
569
570/**
571* Создать HTML для навигации по ревизиям
572*
573* @param  PageChangeLog $ pagelog объект журнала изменений текущей страницы
574* @param  string         $ type     inline или sidebyside
575* @param  int            $ l_rev    левая метка времени ревизии
576* @param  int            $ r_rev    правая метка времени ревизии
577* @возврат строки []html левых и правых элементов навигации
578* @устаревший 2020-07-18
579*/
580 function html_diff_navigation($pagelog, $type, $l_rev, $r_rev)
581 {
582     dbg_deprecated('see ' . PageDiff::class .
'::buildRevisionsNavigation()');
583     return ['', ''];
584 }
```

```
585
586/**
587* Создать HTML-ссылку на разницу, определенную двумя ревизиями
588*
589* @param  string $ difftype тип отображения
590* @param  string $ linktype
591* @param  int $ lrev самая старая версия
592* @param  int $ rrev новейшая ревизия или null для различий с текущей ревизией
593* @return  string html ссылки на разницу
594* @устаревший 2020-07-18
595*/
596 function html_diff_navigationlink($difftype, $linktype, $lrev, $rrev =
null)
597 {
598     dbg_deprecated('see ' . PageDiff::class . '::diffViewlink()');
599     return '';
600 }
601
602/**
603* Вставьте мягкие разрывы в diff html
604*
605* @param  string $ diffhtml
606* @возвращаемая строка
607* @устаревший 2020-07-18
608*/
609 function html_insert_softbreaks($diffhtml)
610 {
611     dbg_deprecated(PageDiff::class . '::insertSoftbreaks()');
612     return (new PageDiff())->insertSoftbreaks($diffhtml);
613 }
614
615/**
616* показывать предупреждение при обнаружении конфликта
617*
618* @автор Андреас Гор <andi@splitbrain.org>
619*
620* @param  строка $ текст
621* @param  string $ резюме
622 * @устаревший 2020-07-18
623*/
624 function html_conflict($text, $summary)
625 {
626     dbg_deprecated(PageConflict::class . '::show()');
627     (new PageConflict($text, $summary))->show();
628 }
629
630/**
631* Выводит глобальный массив сообщений
632*
633* @автор Андреас Гор <andi@splitbrain.org>
634*/
```

```php
function html_msgarea()
{
    global $MSG, $MSG_shown;
    /** @var array $MSG */
    // store if the global $MSG has already been shown and thus HTML output has been started
    $MSG_shown = true;

    if (!isset($MSG)) return;

    $shown = [];
    foreach ($MSG as $msg) {
        $hash = md5($msg['msg']);
        if (isset($shown[$hash])) continue; // skip double messages
        if (info_msg_allowed($msg)) {
            echo '<div class="' . $msg['lvl'] . '">';
            echo $msg['msg'];
            echo '</div>';
        }
        $shown[$hash] = 1;
    }

    unset($GLOBALS['MSG']);
}

/**
* Распечатывает регистрационную форму
*
* @автор Андреас Гор <andi@splitbrain.org>
* @устаревший 2020-07-18
*/
function html_register()
{
    dbg_deprecated(UserRegister::class . '::show()');
    (new UserRegister())->show();
}

/**
* Распечатать форму обновления профиля
*
* @автор Кристофер Смит <chris@jalakai.co.uk>
* @автор Андреас Гор <andi@splitbrain.org>
* @устаревший 2020-07-18
*/
function html_updateprofile()
{
    dbg_deprecated(UserProfile::class . '::show()');
    (new UserProfile())->show();
}

/**
```

```
698* Отображение формы редактирования по умолчанию
699*
700* Это действие по умолчанию для  HTML_EDIT_FORMSELECTION.
701*
702*  @param   массив $ параметр
703* @устаревший 2020-07-18
704*/
691 function html_edit()
692 {
693     dbg_deprecated(Editor::class . '::show()');
694     (new Editor())->show();
695 }
696
697/**
698* Отображение формы редактирования по умолчанию
699*
700* Это действие по умолчанию для  HTML_EDIT_FORMSELECTION.
701*
702*  @param   массив $ параметр
703* @устаревший 2020-07-18
704*/
705 function html_edit_form($param)
706 {
707     dbg_deprecated(Editor::class . '::addTextarea()');
708     (new Editor())->addTextarea($param);
709 }
710
711/**
712* выводит некоторую отладочную информацию
713*
714* @автор Андреас Гор <andi@splitbrain.org>
715*/
716 function html_debug()
717 {
718     global $conf;
719     global $lang;
720     /** @var AuthPlugin $auth */
721     global $auth;
722     global $INFO;
723
724     //remove sensitive data
725     $cnf = $conf;
726     debug_guard($cnf);
727     $nfo = $INFO;
728     debug_guard($nfo);
729     $ses = $_SESSION;
730     debug_guard($ses);
731
732     echo '<html><body>';
733
734     echo '<p>When reporting bugs please send all the following ';
```

```
735     echo 'output as a mail to andi@splitbrain.org ';
736     echo 'The best way to do this is to save this page in your
browser</p>';
737
738     echo '<b>$INFO:</b><pre>';
739     print_r($nfo);
740     echo '</pre>';
741
742     echo '<b>$_SERVER:</b><pre>';
743     print_r($_SERVER);
744     echo '</pre>';
745
746     echo '<b>$conf:</b><pre>';
747     print_r($cnf);
748     echo '</pre>';
749
750     echo '<b>DOKU_BASE:</b><pre>';
751     echo DOKU_BASE;
752     echo '</pre>';
753
754     echo '<b>abs DOKU_BASE:</b><pre>';
755     echo DOKU_URL;
756     echo '</pre>';
757
758     echo '<b>rel DOKU_BASE:</b><pre>';
759     echo dirname($_SERVER['PHP_SELF']) . '/';
760     echo '</pre>';
761
762     echo '<b>PHP Version:</b><pre>';
763     echo phpversion();
764     echo '</pre>';
765
766     echo '<b>locale:</b><pre>';
767     echo setlocale(LC_ALL, 0);
768     echo '</pre>';
769
770     echo '<b>encoding:</b><pre>';
771     echo $lang['encoding'];
772     echo '</pre>';
773
774     if ($auth instanceof AuthPlugin) {
775         echo '<b>Auth backend capabilities:</b><pre>';
776         foreach ($auth->getCapabilities() as $cando) {
777             echo '    ' . str_pad($cando, 16) . ' => ' .
(int)$auth->canDo($cando) . DOKU_LF;
778         }
779         echo '</pre>';
780     }
781
782     echo '<b>$_SESSION:</b><pre>';
783     print_r($ses);
```

```
784     echo '</pre>';
785
786     echo '<b>Environment:</b><pre>';
787     print_r($_ENV);
788     echo '</pre>';
789
790     echo '<b>PHP settings:</b><pre>';
791     $inis = ini_get_all();
792     print_r($inis);
793     echo '</pre>';
794
795     if (function_exists('apache_get_version')) {
796         $apache = [];
797         $apache['version'] = apache_get_version();
798
799         if (function_exists('apache_get_modules')) {
800             $apache['modules'] = apache_get_modules();
801         }
802         echo '<b>Apache</b><pre>';
803         print_r($apache);
804         echo '</pre>';
805     }
806
807     echo '</body></html>';
808 }
809
810/**
811* Форма для запроса нового пароля для существующей учетной записи
812*
813* @автор Бенуа Шено <benoit@bchesneau.info>
814* @автор Андреас Гор <gohr@cosmocode.de>
815* @устаревший 2020-07-18
816*/
817 function html_resendpwd()
818 {
819     dbg_deprecated(UserResendPwd::class . '::show()');
820     (new UserResendPwd())->show();
821 }
822
823/**
824* Верните оглавление, преобразованное в XHTML
825*
826* @автор Андреас Гор <andi@splitbrain.org>
827*
828* @param  массив $ toc
829* @return  string html
830*/
831 function html_TOC($toc)
832 {
833     if ($toc === []) return '';
834     global $lang;
```

```
835     $out  = '<!-- TOC START -->' . DOKU_LF;
836     $out .= '<div id="dw__toc" class="dw__toc">' . DOKU_LF;
837     $out .= '<h3 class="toggle">';
838     $out .= $lang['toc'];
839     $out .= '</h3>' . DOKU_LF;
840     $out .= '<div>' . DOKU_LF;
841     $out .= html_buildlist($toc, 'toc', 'html_list_toc', null, true);
842     $out .= '</div>' . DOKU_LF . '</div>' . DOKU_LF;
843     $out .= '<!-- TOC END -->' . DOKU_LF;
844     return $out;
845 }
846
847/**
848* Обратный вызов для html_buildlist
849*
850* @param  массив $ элемент
851* @return  string html
852*/
853 function html_list_toc($item)
854 {
855     if (isset($item['hid'])) {
856         $link = '#' . $item['hid'];
857     } else {
858         $link = $item['link'];
859     }
860
861     return '<a href="' . $link . '">' . hsc($item['title']) . '</a>';
862 }
863
864/**
865* Вспомогательная функция для создания элементов TOC
866*
867* Возвращает массив, готовый к добавлению в массив TOC
868*
869* @param  string $ link   - куда ссылаться (если $hash установлен в '#', то это
локальный якорь)
870* @param  string $ text   - что отображать в оглавлении
871* @param  int     $ level - уровень вложенности
872* @param  string $ hash   - добавляется к указанной $link, оставьте пустым,
если вам нужны полные ссылки
873* @return  array элемент оглавления
874*/
875 function html_mktocitem($link, $text, $level, $hash = '#')
876 {
877     return  [
878         'link'  => $hash . $link,
879         'title' => $text,
880         'type'  => 'ul',
881         'level' => $level
882     ];
883 }
```

```
884
885/**
886* Вывод объекта Doku_Form.
887* Запускает событие с именем формы: HTML_{$name}FORM_OUTPUT
888*
889* @автор Том Н. Харрис <tnharris@whoopdedo.org>
890*
891* @param  string      $ name Имя формы
892* @param  Doku_Form   $ форма Форма
893* @возврат недействительным
894* @устаревший 2020-07-18
895*/
896 function html_form($name, $form)
897 {
898     dbg_deprecated('use dokuwiki\Form\Form instead of Doku_Form');
899     // Safety check in case the caller forgets.
900     $form->endFieldset();
901     Event::createAndTrigger('HTML_' . strtoupper($name) . 'FORM_OUTPUT',
$form, 'html_form_output', false);
902 }
903
904/**
905* Функция печати формы.
906* Просто вызывает printForm() для объекта формы.
907*
908* @param  Doku_Form $ форма Форма
909* @возврат недействительным
910* @устаревший 2020-07-18
911*/
912 function html_form_output($form)
913 {
914     dbg_deprecated('use ' . Form::class . '::toHTML()');
915     $form->printForm();
916 }
917
918/**
919* Встраивание флэш-объекта в HTML
920*
921* Это создаст необходимый HTML для встраивания флэш-ролика в кроссбраузерный
режим.
922* совместимый способ с использованием допустимого XHTML
923*
924* Параметры $params, $flashvars и $atts должны быть ассоциативными массивами.
925* Для них не нужно делать никакого побега. Альтернативный контент *должен* быть
926* экранировано, потому что используется как есть. Если не указано альтернативное
содержимое
927* Используется $lang['noflash'].
928*
929* @автор Андреас Гор <andi@splitbrain.org>
930* @ссылка    http://latrine.dgx.cz/как-правильно-вставить-вспышку-в-xhtml
931*
```

```
932* @param  string $ swf       - SWF-фильм для встраивания
933* @param  int $ width         - ширина флэш-ролика в пикселях
934* @param  int $ height        - высота флэш-ролика в пикселях
935* @param  array $ params      - дополнительные параметры  (<param>)
936* @param  array $ flashvars - параметры, которые будут переданы в параметре
flashvar
937* @param  array $ atts        - дополнительные атрибуты для тега <object>
938* @param  string $ alt        - альтернативное содержимое (НЕ экранируется
автоматически!)
939* @return  string           - разметка XHTML
940*/
941 function html_flashobject($swf, $width, $height, $params = null,
$flashvars = null, $atts = null, $alt = '')
942 {
943     global $lang;
944
945     $out = '';
946
947     // подготавливаем атрибуты объекта
948     if (is_null($atts)) $atts = [];
949     $atts['width']  = (int) $width;
950     $atts['height'] = (int) $height;
951     if (!$atts['width'])  $atts['width']  = 425;
952     if (!$atts['height']) $atts['height'] = 350;
953
954     // добавить атрибуты объекта для браузеров, соответствующих стандарту
955     $std = $atts;
956     $std['type'] = 'application/x-shockwave-flash';
957     $std['data'] = $swf;
958
959     // добавляем атрибуты объекта для  IE
960     $ie  = $atts;
961     $ie['classid'] = 'clsid:D27CDB6E-AE6D-11cf-96B8-444553540000';
962
963     // открыть объект (с условными комментариями)
964     $out .= '<!--[if !IE]> -->' . NL;
965     $out .= '<object ' . buildAttributes($std) . '>' . NL;
966     $out .= '<!-- <![endif]-->' . NL;
967     $out .= '<!--[if IE]>' . NL;
968     $out .= '<object ' . buildAttributes($ie) . '>' . NL;
969     $out .= '    <param name="movie" value="' . hsc($swf) . '" />' . NL;
970     $out .= '<!--><!-- -->' . NL;
971
972     // параметры печати
973     if (is_array($params)) foreach ($params as $key => $val) {
974         $out .= '  <param name="' . hsc($key) . '" value="' . hsc($val)
. '" />' . NL;
975     }
976
977     // добавить  flashvars
978     if (is_array($flashvars)) {
```

```
979         $out .= '  <param name="FlashVars" value="' .
buildURLparams($flashvars) . '" />' . NL;
980     }
981
982     // альтернативный контент
983     if ($alt) {
984         $out .= $alt . NL;
985     } else {
986         $out .= $lang['noflash'] . NL;
987     }
988
989     // заканчивать
990     $out .= '</object>' . NL;
991     $out .= '<!-- <![endif]-->' . NL;
992
993     return $out;
994 }
995
996/**
997* Печатает HTML-код для заданной структуры вкладки
998*
999* @param  массив  $ вкладки     структура вкладок
1000  * @param string $current_tab the current tab id
1001* @возврат недействительным
1002*/
1003 function html_tabs($tabs, $current_tab = null)
1004 {
1005     echo '<ul class="tabs">' . NL;
1006
1007     foreach ($tabs as $id => $tab) {
1008         html_tab($tab['href'], $tab['caption'], $id === $current_tab);
1009     }
1010
1011     echo '</ul>' . NL;
1012 }
1013
1014/**
1015* Печатает одну вкладку
1016*
1017* @author Катя Арзамасцева <pshns@ukr.net>
1018* @автор Адриан Лэнг <mail@adrianlang.de>
1019*
1020* @param  string $ href - табуляция href
1021* @param  string $ caption - заголовок вкладки
1022* @param  boolean $ selected - выбрана ли вкладка
1023* @возврат недействительным
1024*/
1025
1026 function html_tab($href, $caption, $selected = false)
1027 {
1028     $tab = '<li>';
```

```
1029     if ($selected) {
1030         $tab .= '<strong>';
1031     } else {
1032         $tab .= '<a href="' . hsc($href) . '">';
1033     }
1034     $tab .= hsc($caption)
1035          .  '</' . ($selected ? 'strong' : 'a') . '>'
1036          .  '</li>' . NL;
1037     echo $tab;
1038 }
1039
1040/**
1041* Изменение размера дисплея
1042*
1043* @param  int $ sizechange - размер изменения в байтах
1044* @param  Doku_Form $ form - (необязательно) форма для добавления элементов
1045* @return  void | строка
1046*/
1047 function html_sizechange($sizechange, $form = null)
1048 {
1049     if (isset($sizechange)) {
1050         $class = 'sizechange';
1051         $value = filesize_h(abs($sizechange));
1052         if ($sizechange > 0) {
1053             $class .= ' positive';
1054             $value = '+' . $value;
1055         } elseif ($sizechange < 0) {
1056             $class .= ' negative';
1057             $value = '-' . $value;
1058         } else {
1059             $value = '±' . $value;
1060         }
1061         if (!isset($form)) {
1062             return '<span class="' . $class . '">' . $value .
'</span>';
1063         } else { // Doku_Form
1064             $form->addElement(form_makeOpenTag('span', ['class' =>
$class]));
1065             $form->addElement($value);
1066             $form->addElement(form_makeCloseTag('span'));
1067         }
1068     }
1069 }
1070
```

From:
https://wwoss.ru/ - **worldwide open-source software**

Permanent link:
https://wwoss.ru/doku.php?id=wiki:xref:dokuwiki:inc:html.php

Last update: **2024/08/26 02:35**